Anonimización de Bases de Datos Médicas

Shahad Naji Jaffar y Beatriz Manjón Corrales

GRADO EN INGENIERÍA INFORMÁTICA

FACULTAD DE INFORMÁTICA

UNIVERSIDAD COMPLUTENSE DE MADRID

TRABAJO FIN DE GRADO EN INGENIERÍA INFORMÁTICA

Madrid, 6 de junio del 2016

Director: Rafael Caballero Roldán

AUTORIZACIÓN PARA LA DIFUSIÓN DEL TRABAJO FIN DE GRADO Y SU DEPÓSITO EN EL REPOSITORIO INSTITUCIONAL E-PRINTS COMPLUTENSE

Los abajo firmantes, alumno/s y tutor/es del Trabajo Fin de Grado (TFG) en el Grado en **INGENIERÍA INFORMÁTICA** de la Facultad de **INFORMÁTICA**, autorizan a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor el Trabajo Fin de Grado (TF) cuyos datos se detallan a continuación. Así mismo autorizan a la Universidad Complutense de Madrid a que sea depositado en acceso abierto en el repositorio institucional con el objeto de incrementar la difusión, uso e impacto del TFG en Internet y garantizar su preservación y acceso a largo plazo.

Periodo de embargo (opcional):

- ☐ 6 meses
- ☒ 12meses

TÍTULO del TFG: **ANONIMIZACIÓN DE BASES DE DATOS MÉDICAS**

Curso académico: **2015 / 2016**

Nombre del Alumno/s:
SHAHAD NAJI JAFFAR
BEATRIZ MANJÓN CORRALES

Tutor/es del TFG y departamento al que pertenece:
RAFAEL CABALLERO ROLDÁN
Departamento: SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Firma del alumno/s Firma del tutor/es

Índice

Índice de figuras ..... **IV**

Índice de tablas ..... **V**

Resumen ..... **VI**

Abstract ..... **VII**

Contribuciones de Shahad Naji Jaffar ..... **VIII**

Contribuciones de Beatriz Manjón Corrales ..... **X**

1 | Introducción **12**

1.1 Antecedentes 12
1.2 Nociones de anonimato 12
1.3 Nuestro Objetivo 13

2 | Simulación de Datos **15**

2.1 Población y recursos 16
2.2 Generación aleatoria de citas 18

3 | Vectores de Anonimato **19**

3.1 Insuficiencia del k-anonimato 19
3.2 Definición de vector de anonimato 21

4 | Anonimato como Problema de optimización en Programación con Restricciones **23**

4.1 Programación con Restricciones 23
4.2 El modelo 25
4.3 Posibles mejoras 29

5 | Experimentos **30**

5.1 Mejora de anonimato 30
5.2 Descripción de los experimentos 31

6 | Implementación **36**

7 | Conclusiones y trabajo futuro **40**

8 | Conclusions and future work **42**

9 | Bibliografía **44**

Índice de figuras

Imagen 1: Ejemplo de contenido del fichero .dzn 25

Imagen 2: Primer level 27

Imagen 3: Segundo level 27

Imagen 4: Tercer level 27

Imagen 5: Cuarto level 28

Imagen 6: Resultado final del programa 38

Índice de tablas

Tabla 1: Tabla de población 15

Tabla 2: Tabla de recursos 15

Tabla 3: Tabla de población con asignación de recursos aleatorios 16

Tabla 4: Tabla de población sin asignación de citas aleatorias y mejoradas 17

Tabla 5: Tabla de población con asignación de citas aleatorias 18

Tabla 6: Primer ejemplo de tabla de población con asignación de recursos aleatorios 19

Tabla 7: Segundo ejemplo de tabla de población con asignación de recursos aleatorio 20

Tabla 8: Tercer ejemplo de tabla de población con asignación de recursos aleatorios 20

Tabla 9: Tabla de cuasi-identificadores de los tres ejemplos 21

Tabla 10: Tabla de vectores de anonimato 21

Tabla 11: Primer experimento con población tamaño 10 32

Tabla 12: Segundo experimento con población tamaño 20 33

Tabla 13: Tercer experimento con población tamaño 30 34

Resumen

La publicación de los resultados de programas de screening (exámenes médicos orientados a un grupo genérico de la población) es de sumo interés para la comunidad científica. A pesar de eliminar datos personales como DNI, nombre, etc. el resto de la información, los llamados cuasi-identificadores (código postal, género, edad, profesión, o información de la cita médica incluyendo el centro y la hora) pueden ser utilizados para desvelar la identidad de los participantes en el programa. En particular, la información de la cita médica puede resultar comprometedora si, ya sea intencionadamente o por casualidad, se descubre que una persona determinada ha acudido a un determinado centro en una fecha concreta. Bastaría entonces con consultar los resultados del test para saber si el individuo cuyos cuasi-identificadores conocemos padece la enfermedad. El objetivo de este trabajo es programar la asignación de citas, de manera que se aumente el nivel de anonimato de las bases de datos finales. Para ello, se pretende que personas con características comunes (edad, etc.) acudan a la misma cita (hora y centro), dificultando la identificación de los resultados médicos de un individuo aunque se conozcan sus datos personales, ya que se encontrarán con varias personas con los que comparte los mismos cuasi-identificadores.

Palabras Clave: Bases de datos, Cuasi-identificadores, Datos personales, Generación de citas, Nivel de anonimato, Screening

Abstract

The publication of screening programmes' results (examination or testing of a group of individuals to separate those who are well from those who have an undiagnosed disease or defect or who are at high risk [11]) is an important factor for the scientific community. In spite of deleting personal data such as ID, name, etc. the rest of the information, which is called quasi-identifiers (P.O. Box, gender, age, profession or the information of the appointment including the medical centre and the time) could be used to reveal the identity of the participants of the screening programme. Particularly, the information of the medical appointment may be incriminating, either intentionally or by accident, that disclose a specific person attends a particular centre at a known date. Then, it will be enough with consulting the tests results to know if the individual with the quasi-identifiers, which we know, suffers from illness. The objective of this project is to programme the assignment of the appointments, in order to increase the anonymity level of the final database. Then, the plan is that people with the same characteristics (age, etc) attend the same appointment (time and centre), hinder the identification of the medical results of a specific individual even if his or her personal data are known, because, the information of the same quasi-identifiers of more than one person will be found.

Keywords: Anonymity level, Assignment of the appointments, Database, Personal data, Quasi-identifier, Screening

Contribuciones de Shahad Naji Jaffar

Como Beatriz y yo no solamente somos compañeras que se reunieron para llevar a cabo el trabajo fin de grado, sino también amigas, ambas nos coordinamos de tal manera que podemos decir que el trabajo fue realizado totalmente en conjunto mediante la compartición, el fusionamiento de nuestras ideas y esfuerzo.

Después de cada reunión con el tutor del proyecto cada una preparaba un posible avance sobre lo discutido en dicha reunión, para posteriormente escoger la mejor idea, o elegir la mejor parte de la misma, efectuando un fusionamiento de ideas, por lo que se lograba una mejor con la ayuda de nuestro tutor.

En un primer momento, ambas leímos varios artículos sobre cómo mejorar el anonimato de una tabla, haciendo un gran hincapié sobre el k-anonimato.

La implementación de este proyecto se ha dividido entre cinco programas, cuyo lenguaje de programación es Java, que son invocados mediante un programa en Bash.

Para la instalación de la base de datos relacional que contiene los datos de personas y recursos. En particular hemos elegido PostgreSQL en su versión 1.22.0 Beta 1.

En el primer programa mi labor destaca en la creación de una función escrita PostgreSQL que asegura que los recursos pueden atender a toda la población por lo que se aumenta la capacidad de cada recurso hasta la máxima capacidad establecida por el usuario final. Si no se consigue que los recursos cubran a toda la población, el programa se detendrá invocando a una excepción.

Para asignar las citas a la población tanto mi compañera Beatriz como yo nos reunimos para programar el segundo programa, usando la función Random del paquete java.util.Random.

Durante el desarrollo del proyecto se ha querido saber cuál es el valor del k-anonimato de las tablas eligiendo ciertos atributos que representan a los cuasi-identificadores. Para ello, Beatriz y yo; y bajo la supervisión de nuestro tutor creamos el tercer programa.

Cabe destacar que este tercer programa nos aportó tanto a mi compañera como a mí un mejor entendimiento de este Trabajo Fin de Grado refinando los conceptos.

Ambas implementamos funciones necesarias del cuarto programa para calcular: Q, f, R y c.

Para obtener los valores de los datos anteriores, creamos varias funciones que acceden a las tablas de la base de datos PostgreSQL tras comprobar la existencia de las mismas.

En este programa, destaca la labor de mi compañera Beatriz por la correcta colocación de los parámetros Q, f, R y c; y sus respectivos valores en el fichero con extensión .dzn para que estos sean correctamente leídos por el programa de MiniZinc, de aquí en adelante appointment.mzn.

Para la obtención del vector anonimato, es decir para conseguir una mejora de anonimato se ha procedido a programar el quinto programa.

Debemos destacar la labor de nuestro tutor con respecto a la implementación de appointment.mzn y además del cálculo de distancia entre los distintos vectores generados, es decir, la medida de mejora de anonimato obtenida.

Antes de proceder a ejecutar appointment.mzn se debe realizar una lectura del fichero con extensión .dzn guardando los valores Q, f, R y c de en variables según su tipo apropiado para su posterior uso en este mismo programa. Aquí destaca mi labor junto con la supervisión de mi compañera Beatriz que antes de la ejecución de appointment.mzn realizó varias funciones que calculan el vector de anonimato base y el vector de anonimato aleatorio escribiéndolos en el fichero .dzn para su posterior comparación con el vector que se generará a continuación por MiniZinc.

Después de esta lectura, además mi labor se ha centrado en la llamada a MiniZinc a partir de Java, y actualizar los valores del fichero .dzn con los valores devueltos por appointment.mzn.

Debemos mencionar que gracias a este Trabajo Fin de Grado hemos expandido nuestros conocimientos utilizando PostgreSQL por primera vez, así como la programación con restricciones (MiniZinc) y la implementación de un programa en Bash Shell.

Para la elaboración del programa Bash Shell, nos reunimos para realizar dos versiones, una en español y otra en inglés, refinándolos durante el desarrollo de los experimentos.

Cabe destacar, la gran labor de mi compañera Beatriz en la realización de los experimentos a partir de los parámetros establecidos por todo el equipo de este Trabajo Fin de Grado (el tutor y nosotras dos).

Para el posterior análisis de los experimentos, elaboramos varias tablas para comparar los distintos valores que se generaron a partir de toda la implementación anterior y poder llegar a ciertas conclusiones.

Finalmente, la escritura de esta memoria se ha realizado totalmente en conjunto entre los componentes del equipo que ha desarrollado este Trabajo Fin de Grado.

Contribuciones de Beatriz Manjón Corrales

Como Shahad y yo no solamente somos compañeras que se reunieron para llevar a cabo el trabajo fin de grado, sino también amigas, ambas nos coordinamos de tal manera que podemos decir que el trabajo fue realizado totalmente en conjunto mediante la compartición, el fusionamiento de nuestras ideas y esfuerzo.

Después de cada reunión con el tutor del proyecto cada una preparaba un posible avance sobre lo discutido en dicha reunión, para posteriormente escoger la mejor idea, o elegir la mejor parte de la misma, efectuando un fusionamiento de ideas, por lo que se lograba una mejor con la ayuda de nuestro tutor.

En un primer momento, ambas leímos varios artículos sobre cómo mejorar el anonimato de una tabla, haciendo un gran hincapié sobre el k-anonimato.

La implementación de este proyecto se ha dividido entre cinco programas, cuyo lenguaje de programación es Java, que son invocados mediante un programa Bash.

Para la instalación de la base de datos relacional que contiene los datos de personas y recursos. En particular hemos elegido PostgreSQL en su versión 1.22.0 Beta 1.

En el primer programa, por mi parte me encargué de la creación de las tablas y de la conexión entre el programa y PostgreSQL, a partir de un ejemplo de código que nos envió el tutor. Asimismo, tanto mi compañera Shahad como yo acordamos que mi idea para la generación de la información aleatoria de la población era la más óptima, ya que damos la opción al usuario de introducir todos los datos que desee sin ninguna limitación, es decir, el usuario no elige desde unos parámetros preestablecidos en el programa, sino que él mismo introduce los valores que quiera.

Para asignar las citas a la población, ambas nos reunimos para programar el segundo programa, usando la función Random del paquete java.util.Random.

Durante el desarrollo del proyecto se ha querido saber cuál es el valor del k-anonimato del eligiendo ciertos atributos que representan a los cuasi-identificadores. Para ello, Shahad y yo; y bajo la supervisión de nuestro tutor creamos el tercer programa.

Cabe destacar que este tercer programa nos aportó tanto a mi compañera como a mí un mejor entendimiento de este Trabajo Fin de Grado, refinando los conceptos.

Ambas implementamos las funciones necesarias del cuarto programa para calcular: Q, f, R y c. Para obtener los valores de los datos anteriores, creamos varias funciones que acceden a las tablas de la base de datos PostgreSQL tras comprobar la existencia de las mismas.
En este programa, destaca mi labor a la hora de la correcta colocación de los parámetros Q, f, R y c; y sus respectivos valores en el fichero con extensión .dzn para que estos sean correctamente leídos por el programa de MiniZinc, de aquí en adelante appointment.mzn.

Para la obtención del vector anonimato, es decir para conseguir una mejora de anonimato se ha procedido a programar el quinto programa.

Debemos destacar la labor de nuestro tutor con respecto a la implementación de appointment.mzn y además del cálculo de distancia entre los distintos vectores generados, es decir, la medida de mejora de anonimato obtenida.

Antes de proceder a ejecutar appointment.mzn se debe realizar una lectura del fichero con extensión .dzn guardando los valores Q, f, R y c de en variables según su tipo apropiado para su posterior uso en este mismo programa. Mi parte de trabajo ha consistido en, antes de la ejecución de appointment.mzn, realizar varias funciones que calculan el vector de anonimato base y el vector de anonimato aleatorio escribiéndolos en el fichero .dzn para su posterior comparación con el vector que se generará a continuación por MiniZinc.

Después de esta lectura, he ayudado a mi compañera Shahad aportando mis ideas para programar el resto del código, es decir, la llamada a MiniZinc a partir de Java, y actualizar los valores del fichero .dzn con los valores devueltos por appointment.mzn.

Tras la realización de varios experimentos he llegado a la conclusión de que el resolutor base de MiniZinc ralentizaba el tiempo de ejecución por lo que tanto mi compañera como yo optamos por usar otro resolutor bajo la aprobación de nuestro tutor.

Debemos mencionar que gracias a este Trabajo Fin de Grado hemos expandido nuestros conocimientos utilizando PostgreSQL por primera vez, así como la programación con restricciones (MiniZinc) y la implementación de un programa en Bash Shell.

Para la elaboración del programa Bash Shell, nos reunimos para realizar dos versiones, una en español y otra en inglés, refinándolos durante el desarrollo de los experimentos.

Debido a problemas de compatibilidad entre programas y sistemas operativos, me he encargado de la realización de los experimentos a partir de los parámetros establecidos por todo el equipo de este Trabajo Fin de Grado (el tutor y nosotras dos).

Finalmente, la escritura de esta memoria se ha realizado totalmente en conjunto entre los componentes del equipo que ha desarrollado este Trabajo Fin de Grado.

1 | Introducción

1.1 Antecedentes

Hoy en día, los datos personales de individuos se encuentran registrados en bases de datos pertenecientes a distintos organismos (Hospitales, Padrón, Colegios, Universidades, etc).

Sin embargo, a partir de distintas publicaciones realizadas por dichos organismos, la privacidad puede llegar a verse comprometida. El enlazamiento de los atributos de dichas publicaciones, permite averiguar, en ocasiones, los datos referentes a un único individuo.

Cuando, a partir del análisis de datos no sensibles (lo que llamaremos a continuación cuasi-identificadores), como el código postal, la edad y la profesión de una tabla médica se llega a determinar a un único individuo del censo, decimos que obtenemos información sensible. En ocasiones, en lugar de identificar de forma única un individuo, la información puede servir para indicar con mayor probabilidad ciertas características de un segmento de la población. Por ejemplo, si sabemos por unos datos que las personas de una cierta edad sufren más de una enfermedad, y que por otros datos dichas personas se concentran en ciertos barrios.

Esto ha dado lugar a un interés creciente por el concepto de anonimato de una base de datos pública, así como al estudio de técnicas que permitan incrementar dicho anonimato.

1.2 Nociones de anonimato

Casi todas las nociones de anonimato se basan en el concepto de cuasi-identificador. Dentro de una tabla de datos personales, hay algunos datos (atributos) que identifican perfectamente a cada individuo: DNI, número de la seguridad social, etc. En el lenguaje de las bases de datos se les llama clave o identificador. En las tablas que se publican y que contienen información que se debe proteger (diagnósticos, por ejemplo), los identificadores se eliminan para proteger el anonimato.

Sin embargo, las tablas contienen otros atributos, que, si se publican, o al menos se comparten entre centros médicos, como la edad, el género, la profesión que, al combinarlos entre sí, pueden acabar identificando a un único individuo, o al menos indicar que cierto individuo pertenece a un grupo de tamaño reducido. Estos atributos son llamados cuasi-identificadores. Hay que tener en cuenta que un “atacante” puede conocer con facilidad los cuasi-identificadores de una persona, y que, en particular, si estos datos no se repiten en una base de datos médicas, conlleva la identificación de su diagnóstico.

Entre estos cuasi-identificadores se pueden contar los datos de la cita médica: día, hora y centro médico. El “atacante” puede conocer esta información, ya sea porque vigile a la persona, o porque sea miembro de la institución médica, o incluso un trabajador del centro médico.

Por otra parte, la publicación de estos datos es importante porque en ocasiones el lugar de la cita (diferentes centros médicos con diferentes recursos y equipos de trabajo) o incluso la hora puede introducir sesgos que es preciso determinar.

K-ANONIMATO

Samarati y Sweeney [1,2] propusieron una definición de privacidad llamada k-anonimato. Dados un conjunto de cuasi-identificadores Q, se dice que una tabla satisface un nivel k de anonimato, o que verifica el k-anonimato, si cada registro de la tabla es indistinguible de al menos k-1 de los otros registros con respecto a Q.

En otras palabras, si se verifica el k-anonimato, sabemos que cada cuasi-identificador se repite al menos en k registros de la tabla, asegurando, para valores elevados de k, que los individuos no puedan ser identificados fácilmente.

PROGRAMAS DE SCREENING

El screening se puede definir como la aplicación de procedimientos de selección (cuestionario, examen físico, test) a poblaciones de individuos aparentemente «sanos» con objeto de identificar a aquellos que pueden estar enfermos o que presentan un riesgo incrementado de padecer una determinada enfermedad porque presentan un cierto factor de riesgo. Se trata de diferenciar a las personas aparentemente sanas que probablemente padecen una enfermedad o presentan un factor de riesgo de las aparentemente sanas que probablemente no padecen la enfermedad o no presentan el factor de riesgo [7].

Al inicio del programa de screening se identifica al segmento de la población al cual se desea someter a diferentes pruebas para detectar posibles enfermedades con antelación. A partir de esta identificación, se generarán citas que incluirán tanto el lugar como la fecha para cada individuo. Es importante señalar que las citas son elegidas sin participación del individuo, al que se comunica la cita. Esto facilita una adecuada elección de citas, por ejemplo, para aumentar el anonimato, lo que no sucede con las citas concertadas directamente por los pacientes. Una vez que el segmento de población se ha sometido a una serie de análisis en sus respectivas citas, se obtienen resultados que serán compartidos por distintas instituciones médicas.

Como se ha mencionado anteriormente, el segmento de población inicial se guarda en una base de datos. Esta base de datos ya tiene de por sí un cierto nivel de anonimato al que denominamos anonimato base. Al asignar la cita cada individuo de la población se obtiene un nuevo nivel de anonimato, que siempre será igual o peor que el anonimato base.

1.3 Nuestro Objetivo

La asignación aleatoria de citas, que se realiza habitualmente en los programas de screening, tiene como efecto una gran pérdida de anonimato con respecto al anonimato base. Nuestro objetivo en este trabajo es realizar una asignación de citas médicas en programas de screening, que minimice esta pérdida de anonimato.

Durante el desarrollo del trabajo hemos observado que la noción de k-anonimato tal y como se encuentra en la literatura, no es suficientemente precisa: instancias de base de datos con una clara diferencia en cuanto a anonimato, reciben sin embargo el mismo valor k. Por ello definimos un nuevo concepto que refina el k-anonimato, los vectores de anonimato.

Hay que señalar que la mayor parte de las técnicas para incrementar el k-anonimato afectan a la calidad de los datos ya que se basan en la modificación, supresión o adición de filas.

En nuestro caso se pretende diseñar un sistema de citas que garantice la menor pérdida de anonimato sin pérdida de calidad de los datos.

Para ello definiremos el problema de la asignación de citas, como un problema de optimización, empleando para resolver programación con restricciones.

2 | Simulación de Datos

Para poner a prueba nuestra propuesta, precisamos un conjunto de datos que represente una población, con sus cuasi-identificadores, así como ejemplos de recursos que engloban tanto el centro médico como sus citas. Sin embargo, los datos médicos son confidenciales, por lo que no se puede acceder a poblaciones usadas en casos reales de screening. Por ello, para la realización de este proyecto se introduce un mecanismo que permite simular un segmento de población y una serie de recursos.

A continuación, se expondrá un ejemplo de una población ficticia (grupo de individuos) consistente en un segmento de población de 15 personas, con edades comprendidas entre los 16 y los 18 años y de ambos géneros. Además, se dispone de cinco recursos cuyas capacidades oscilan entre dos y cuatro personas.

ID	Edad	CP	Género	Número de Profesiones
1	16	2	0	1
2	17	1	0	1
3	16	1	1	1
4	16	2	1	1
5	17	2	0	1
6	16	2	1	1
7	17	2	1	1
8	16	1	1	1
9	16	2	0	1
10	16	2	0	1
11	18	1	1	1
12	16	2	1	1
13	16	1	0	1
14	18	2	0	1
15	18	1	1	1

Tabla 1

Recurso	Capacidad
1	2
2	4
3	3
4	3
5	3

Tabla 2

Edad	CP	Género	Número de Profesiones	Recurso Aleatorio
16	2	0	1	3
17	1	0	1	3
16	1	1	1	1
16	2	1	1	5
17	2	0	1	1
16	2	1	1	4
17	2	1	1	3
16	1	1	1	4
16	2	0	1	5
16	2	0	1	2
18	1	1	1	4
16	2	1	1	2
16	1	0	1	2
18	2	0	1	2
18	1	1	1	5

Tabla 3

Como se puede observar en la *tabla 1***,** los campos que son: ID, que representa el nombre y apellidos del individuo; su edad; su Código Postal (CP) y su profesión.

La *tabla 2* se compone de dos campos; recurso, que es una representación del hospital y la hora de la cita, y la capacidad de cada recurso, es decir, cuantos individuos pueden acudir a un recurso concreto.

En la *tabla 3*, se elimina el campo ID, que identifica el nombre y apellidos del individuo para garantizar cierto anonimato. Estos son los datos iniciales de los que se parte. Se puede comprobar que el anonimato base es: (5, 2, 2).

Así mismo se añade un nuevo campo, Recurso Aleatorio, que hace referencia a uno de los recursos de la *tabla 2*. Para el interés de la comunidad científica, se pretende publicar no sólo los datos de la población, sino también las citas generadas expuestas en la *tabla 3*, conservando cierto anonimato.

El siguiente apartado explica el mecanismo que permite generar de forma aleatoria tanto la población como sus recursos. Posteriormente, se explicará de qué manera se asignan los recursos a cada individuo de la población, simulando así una generación aleatoria de citas.

2.1 Población y recursos

Para la representación de un segmento de población específico, se establecen los siguientes campos:

En primer lugar, se indica la porción de población que se quiere estudiar, así como el rango de edades, es decir, edad mínima y máxima de los pacientes que se van a examinar; también se incluye la cantidad de códigos postales para determinar la zona en la que vive la población, el género de la población, representados por: 0 para mujer, 1 para hombre y 2 refiriéndose a ambos

géneros. Finalmente, también parametrizamos la población con el número de profesiones, que representa la actividad que desempeña cada individuo.

Así mismo, se debe detallar la cantidad de recursos disponibles, indicando la capacidad mínima y máxima que puede tener cada recurso, asegurando que estos recursos pueden cubrir a toda la población creada.

Todos estos datos, tanto los de la población como los de los recursos, son indicados por el usuario de nuestra aplicación, lo que nos permite realizar simulaciones de diversas situaciones.

Edad	CP	Género	Recurso Normal	Recurso Aleatorio
16	2	0		
17	1	0		
16	1	1		
16	2	1		
17	2	0		
16	2	1		
17	2	1		
16	1	1		
16	2	0		
16	2	0		
18	1	1		
16	2	1		
16	1	0		
18	2	0		
18	1	1		

Tabla 4*

Para almacenar la población generada a partir de los parámetros descritos anteriormente, se hace uso de PostgreSQL, un sistema de gestión de bases de datos relacional. Un ejemplo de tabla de población generada puede verse en la *tabla 4*. Nótese la existencia de dos columnas vacías, recurso aleatorio y recurso normal. El primero simulará la asignación de citas aleatoria, como veremos en este mismo capítulo. El segundo se generará utilizando programación con restricciones, de esta forma podremos establecer una comparativa entre ambos métodos.

Igualmente, se generará una tabla detallando la información de los recursos, como se puede observar en la *tabla 2*. Todos estos datos se obtienen generando números aleatorios a partir de los parámetros iniciales fijados por el usuario, siempre asegurando a que los recursos generados cubren a todos los individuos maximizando la capacidad de cada recurso hasta que se asegure que se atenderá a todo el segmento de población.

*Se ha eliminado el campo “Número de profesiones” ya que tiene el mismo valor para todos los individuos.

2.2 Generación aleatoria de citas

Tras generar la información del segmento de población a examinar, y detallar los recursos de los que se dispone, se procede a asignar un recurso a cada individuo de la población de forma aleatoria; además de que no se supere la capacidad máxima de cada recurso y que se asegure una cita para cada individuo.

ID	Edad	CP	Género	Número de Profesiones	Recurso Normal	Recurso Aleatorio
1	16	2	0	1		3
2	17	1	0	1		3
3	16	1	1	1		1
4	16	2	1	1		5
5	17	2	0	1		1
6	16	2	1	1		4
7	17	2	1	1		3
8	16	1	1	1		4
9	16	2	0	1		5
10	16	2	0	1		2
11	18	1	1	1		4
12	16	2	1	1		2
13	16	1	0	1		2
14	18	2	0	1		2
15	18	1	1	1		5

Tabla 5

En la *tabla 5* se muestra la asignación de los recursos de forma aleatoria a cada uno de los individuos. Obsérvese que se respetan los límites de las capacidades máximas de los recursos iniciales.

3 | Vectores de Anonimato

3.1 Insuficiencia del k-anonimato

Como se ha explicado en el capítulo 1, en particular dentro del apartado Nociones de Anonimato, existen atributos a los que denominamos cuasi-identificadores; mediante los cuales se puede llegar a revelar la identidad de un individuo, o incluso su diagnóstico.

Por ejemplo, en las tablas del capítulo anterior se han considerado como cuasi-identificadores a los atributos edad, código postal y género. En este capítulo de momento no consideramos como cuasi-identificador la cita (la asignación de un individuo a un recurso).

El objetivo de este capítulo es mostrar que el concepto de k-anonimato no refleja de forma suficientemente precisa el anonimato de una población.

Para ejemplificar esto, se exponen a continuación tres ejemplos de poblaciones obtenidos a partir de los mismos parámetros iniciales:

Edad	CP	Género	Número de Profesiones	Recurso Aleatorio
16	2	0	1	3
17	1	0	1	3
16	1	1	1	1
16	2	1	1	5
17	2	0	1	1
16	2	1	1	4
17	2	1	1	3
16	1	1	1	4
16	2	0	1	5
16	2	0	1	2
18	1	1	1	4
16	2	1	1	2
16	1	0	1	2
18	2	0	1	2
18	1	1	1	5

Tabla 6

Edad	CP	Género	Número de Profesiones	Recurso Aleatorio
17	2	1	1	5
18	2	1	1	3
16	2	1	1	1
18	2	0	1	3
17	1	1	1	3
17	1	0	1	2
18	1	1	1	4
17	2	1	1	4
17	1	1	1	2
18	2	0	1	5
17	2	1	1	5
18	1	0	1	5
17	1	0	1	1
18	2	1	1	1
18	1	0	1	1

Tabla 7

Edad	CP	Género	Número de Profesiones	Recurso Aleatorio
18	2	1	1	2
16	2	0	1	2
17	1	0	1	5
16	1	0	1	5
18	2	1	1	3
18	1	1	1	4
16	1	0	1	5
17	2	0	1	2
16	1	0	1	3
16	2	0	1	1
18	2	1	1	1
18	2	1	1	1
17	2	0	1	4
18	1	0	1	4
18	2	1	1	4

Tabla 8

Analizando las tablas se comprueba que:

- En la *tabla 6* se verifica que el cuasi-identificador **(edad=17, código postal=2, género=1)** (que a partir de ahora se escribe de forma abreviada con la tupla (17, 2, 1)), solo aparece una vez.
- En la *tabla 7* el cuasi-identificador **(18, 1, 1)** aparece igualmente una sola vez.
- Análogamente, en la *tabla 8* el cuasi-identificador **(18, 1, 1)** aparece una sola vez.

De aquí se deduce que, en las tres pruebas realizadas, se verifica un nivel de 1-anonimato. Sin embargo, no se puede asumir que las tres tablas tienen el mismo nivel de anonimato. Es verdad que en las tres hay al menos un individuo cuyo cuasi-identificador se repite solo una vez, pero en la *tabla 6* se pueden identificar otras a cinco personas en la misma situación, mientras que en las *tablas 7* y *8* se pueden identificar a dos y tres personas respectivamente.

Por consiguiente, parece lógico concluir que la *tabla 7* tiene mayor nivel de anonimato, porque tiene "solo" a dos personas en riesgo, frente a las tres o cinco personas cuya identidad podría ser desvelada en los otros casos.

Se necesita por tanto una mejor definición, que tenga en cuenta no solo el nivel de k-anonimato sino **cuántas** personas están en ese nivel. Esta idea nos llevará al concepto de vector de anonimato.

3.2 Definición de vector de anonimato

El vector de anonimato cuenta el número veces que se repite cada valor que toma el cuasi-identificador en la tabla. Esta información se representa en forma de vector, de tal manera que por ejemplo la posición 1 del vector contiene el número de cuasi-identificadores que se repiten una sola vez. Análogamente, la posición 2 cuenta el número de cuasi-identificadores con dos repeticiones, y así sucesivamente.

En la siguiente tabla se listan los distintos valores de cuasi-identificadores que aparecen en las tres tablas del ejemplo, además de las veces que aparecen repetido cada cuasi-identificador en cada tabla:

Q	Tabla 6 (=9)	Tabla 7(=8)	Tabla 8(=7)
q1 = (16,2,0)	3	0	2
q2 = (17,1,0)	1	2	1
q3 = (16,1,1)	2	0	0
q4 = (16,2,1)	3	1	0
q5 = (17,2,0)	1	0	2
q6 = (17,2,1)	1	3	0
q7 = (18,1,1)	2	1	1
q8 = (16,1,0)	1	0	3
q9 = (18,2,0)	1	2	0
q10 = (18,2,1)	0	2	5
q11= (17,1,1)	0	2	0
q12 = (18,1,0)	0	2	1

Tabla 9

Por ejemplo, nótese que en la *tabla 6* tiene, como se ha dicho anteriormente, 5 cuasi-identificadores con una sola repetición, concretamente q2, q5, q6, q8 y q9. A partir de esta idea, la *tabla 9* se puede representar de forma abreviada mostrando cuántas veces aparece cada repetición del cuasi-identificador en cada prueba:

Tabla \ k	1	2	3	4	5
Tabla 6	5	2	2	0	0
Tabla 7	2	5	1	0	0
Tabla 8	3	2	1	0	1

Tabla 10

La filas de la tabla corresponden directamente con los vectores de anonimato: (5,2,2,0,0) para la *Tabla 6*, (2,5,1,0,0) para la *Tabla 7*, y (3,2,1,0,1) para la *Tabla 8*.

Algunas propiedades de estos vectores:

P1. El k-anonimato corresponde a la posición del primer valor distinto de 0 empezando por la izquierda. En el ejemplo las tres tablas verifican k=1.

P2. Los ceros a la derecha no cuentan, es decir se puede considerar (5, 2, 2, 0, 0) y (5, 2, 2) el mismo vector de anonimato. En general llamamos longitud *l* de un vector de anonimato *v* a la posición más a la derecha que tenga un valor distinto de 0. En el ejemplo *v* = (5, 2, 2, 0, 0) hablaríamos de una longitud *l* = 3.

P3. Dada una población de tamaño *n*, con un vector *v* de longitud *l*, se cumple que:

- sum (i = 1, i = l) v[i] * i = n.

Por ejemplo, el vector *v* = (5, 2, 2, 0, 0) corresponde a una población de 5*1+2*2+2*3 = 15 (en otras palabras 5 valores del cuasi-identificador se repiten una vez, dos valores se repiten dos veces, y otros dos valores del cuasi-identificador aparecen tres veces).

Ahora resulta fácil comparar vectores de anonimato, siguiendo el orden lexicográfico que se representa por < . En este caso se tiene:
(2,5,1,0,0) < (3,2,1,0,1) < (5,2,2,0,0)

P4. Un vector menor significa más anonimato y por tanto ahora podemos decir que la tabla 7 representa un mejor (mayor) anonimato. La cuarta propiedad indica que nuestros vectores son un refinamiento del concepto de k-anonimato.

P5. Sean v1, v2 dos vectores de anonimato para dos tablas T1, T2 de tamaño *n*, tales que T1 verifica k1-anonimato y T2 k2-anonimato. Entonces, si v1<v2, se cumple k1 >= k2.

Así mismo, mediante la columna de recursos aleatorios, se generan nuevos vectores de anonimato que informan del reparto aleatorio de las citas, de aquí en adelante llamados “vector aleatorio”.

Los vectores aleatorios de las *tablas 6, 7* y *8* son respectivamente (15), (13,1) y (11,2).

A partir de dichos vectores, los vectores iniciales de la *tabla 10* y teniendo en cuenta lo expuesto en la propiedad 3 (P3), se observa que el anonimato empeora significativamente.

Nótese, por ejemplo, que en el vector inicial correspondiente a la *tabla 6* (5, 2, 2) se pasa de identificar 5 personas a poder identificar hasta 15 personas lo que conlleva a una gran pérdida de anonimato.

También se genera una pérdida de anonimato a la hora de incluir nuevos atributos que identifican a los cuasi-identificadores, ya que aumentará su variación, es decir que se vería la *tabla 9* más extendida.

4 | Anonimato como Problema de optimización en Programación con Restricciones

4.1 Programación con Restricciones

Para obtener el mejor vector de anonimato posible se utiliza el paradigma de programación conocido como programación con restricciones.

La programación con restricciones es un estilo de programación que agrupa a muchos lenguajes. Se puede encuadrar dentro del paradigma de la programación declarativa, donde el programador indica qué quiere conseguir sin detallar los pasos que hay que seguir para lograrlo puesto que el sistema es el que dispone de forma interna de los métodos que utilizarán para lograr el objetivo propuesto por el programador. Además, la programación con restricciones y la programación lógica se combinan de forma natural en la llamada programación lógica con restricciones (Constraint Logic Programming o paradigma CLP) [3].

En programación con restricciones se definen variables sobre un cierto dominio (por ejemplo, enteros, booleanos, reales, etc.), y se indican las relaciones que deben cumplir las variables entre sí. El resultado es un programa al que se le suele llamar modelo.

Por ejemplo, usando notación del lenguaje de restricciones MiniZinc [9] podemos definir modelos como el siguiente:

```
%%%%%%%%%%%%%%%%%
var 0..10:x;
var 0..5:y;
constraint x>y;
constraint x+y > 11;
solve satisfy;
%%%%%%%%%%%%%%%%%
```

El programa, en este caso MiniZinc, usa resolutores adecuados para encontrar valores de las variables que satisfagan el modelo. En este caso encuentra la solución:

```
x = 10;
y = 2;
```

En programación con restricciones no solo se plantean problemas de satisfacción como el anterior; también se pueden definir problemas de optimización:

```
%%%%%%%%%%%%%%%%%%%
var 0..10:x;
var 0..5:y;
constraint x > 1;
constraint x+y>=11;
solve maximize x;
%%%%%%%%%%%%%%%%%%%
```

En este caso buscamos el mayor valor de x que satisface el modelo, y el resolutor nos devuelve las siguientes posibles soluciones:

```
x = 10;
y = 1;
----------
```

Se va a modelar el problema de anonimato como un problema de optimización en el lenguaje MiniZinc.

4.2 El modelo

Vamos a obtener el vector de anonimato de forma iterativa; cada vez que se aplique el modelo se obtendrán un nuevo componente del vector. Para ello definimos un modelo que consta de dos partes:

- El modelo en sí, que busca minimizar la posición *l* del vector.
- Un fichero de parámetros con extensión .dzn que va variando, indicando el valor de *l*, y los valores anteriores ya encontrados.

Las iteraciones son controladas desde un programa Java, que va modificando el fichero de parámetros hasta completar un vector de anonimato óptimo que cubra toda la población.

Para explicar el modelo el primer paso es definir los parámetros que se van a usar:

- *Q:* número de cuasi-identificadores diferentes.
- *f:* vector que representa la frecuencia de cada cuasi-identificador, es decir, el número de veces que se repite cada cuasi-identificador.
- *R:* su valor representa el número de recursos de los que se dispone.
- *c:* vector de capacidades de cada recurso.

Un ejemplo de dzn inicial al que se irán añadiendo valores cada vez que se obtenga un resultado del programa MiniZinc, de aquí en adelante appointment.mzn, se muestra en la siguiente imagen:

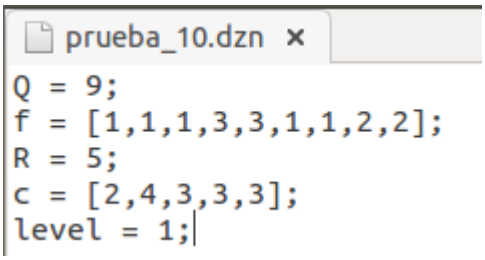

Imagen 1

Ahora podemos ver el modelo MiniZinc:

```
array [1..Q] of int: f;
int:R;
array [1..R] of int: c;
int:level;


array [1..level] of var int:anom;

array [1..Q,1..R] of var int:a;

%C1 initial domain in the the appointments array
constraint forall([a[i,j]>=0| i in 1..Q, j in 1..R]);

% C2 every person is appointed
constraint forall([ sum([a[i,j] | j in 1..R])==f[i] | i in 1..Q]);

% C3 take into account the maximum capacity of each resource
constraint forall([ sum([a[i,j] | i in 1..Q])<=c[j] | j in 1..R]);

% C4 for all the values less than level, ensure that the number of repetitions
% is the already computed
constraint forall([ sum([bool2int(a[i,j]=l)| i in 1..Q, j in 1..R])=anom[l] | l in 1..level]);

% minimize the anonymity vector at position 'level'
solve minimize anom[level];

% show the result
output(["level: "++show(level), "\n anom[level] = ", show(anom[level]),"\n"]++
[show(a[i,j])++(if j==R then "\n" else " " endif) | i in 1..Q, j in 1..R]);
```

En appointment.mzn, además de las variables cuyos valores definidos por el fichero dzn, se declaran las siguientes variables:

- *a*: vector bidimensional que sus filas representan los cuasi-identificadores (*Q*) y sus columnas representan los diferentes recursos de los que dispone (*R*).
- *anom*: vector lineal con el reparto optimizado de las citas.

Por cada iteración del programa de Java se irán recibiendo de appointment.mzn una solución de reparto mejorando en lo posible el k-anonimato hasta cubrir a toda la población aumentando el nivel del vector de anonimato a resolver.

A continuación, se muestra la solución devuelta por appointment.mzn usando en un principio los datos de la *imagen 1* que se modificarán en las siguientes iteraciones del programa de Java:

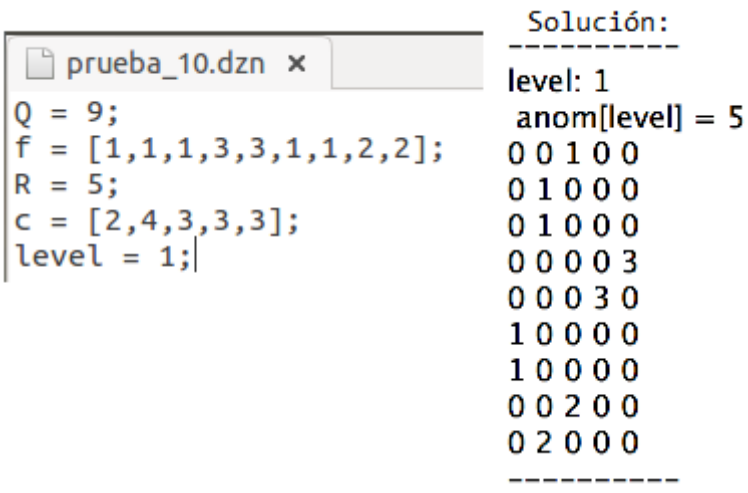

Imagen 2

Debido a que el vector bidimensional a de la *imagen 2* contiene cinco cuasi-identificadores asignados cinco individuos diferentes, lo que conlleva a desvelar la identidad de estos cinco individuos, y se determina que el k-anonimato de este vector de anonimato es 1.

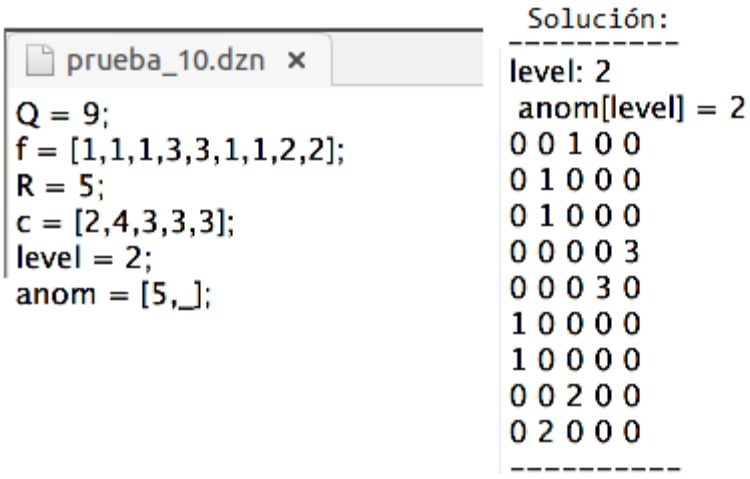

Imagen 3

Así mismo, en la segunda iteración (level 2), se busca cuántas veces se repite el level 2 en el vector *a*, que en el caso de la *imagen 3*, level se repite dos veces.

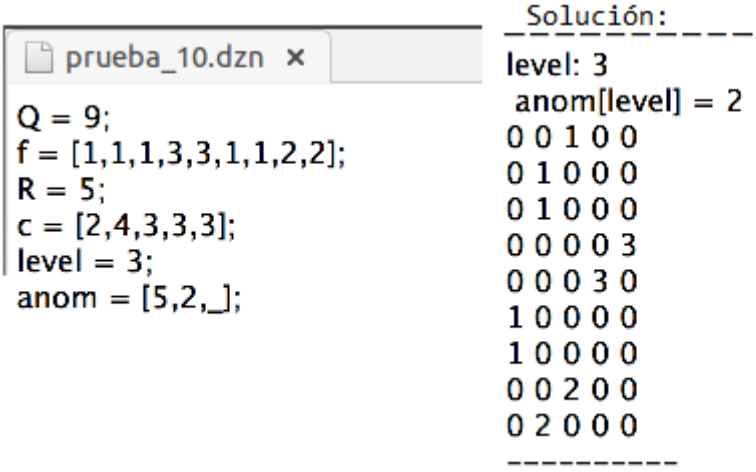

Imagen 4

En la *imagen 4* se observa que el level 3, se repite dos veces (anom[level] = 2), por lo que el vector de anonimato actual de Java es (5,2,2).

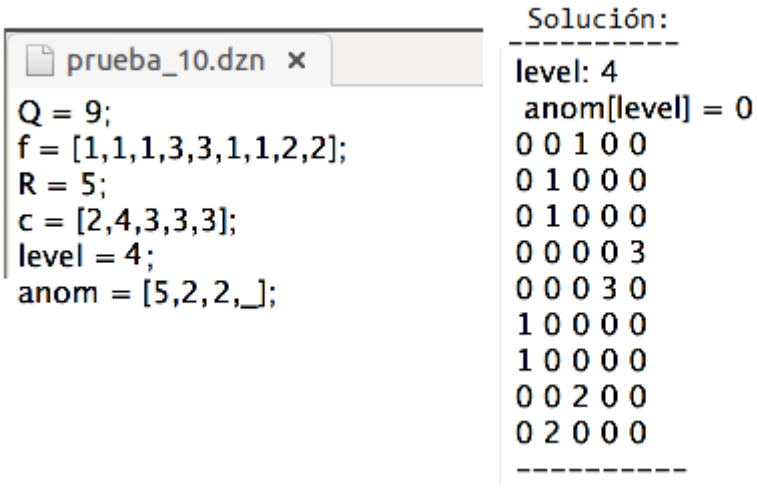

Imagen 5

Para finalizar, el programa Java ejecuta por última vez appointment.mzn con level 4 para comprobar que todos los individuos de la población tengan asignada una cita. Por ello, en este ejemplo, appointment.mzn devuelve 0 porque ya se ha cubierto a toda la población.

En otros ejemplos appointment.mzn devolvería 0 si no existe el valor de este level en ninguna posición del vector bidimensional *a*.

Se puede concluir que el vector de anonimato final es (5, 2, 2).

4.3 Posibles mejoras

Hay que notar que la restricción C4 que se encarga de comprobar que el número de ocurrencias de los valores 1… *l*-1 se pueden establecer mediante una restricción de cardinalidad global [4].

Otra posible mejora sería utilizar los parámetros de búsqueda en MiniZinc para encontrar el mejor vector sin iteraciones ni programa, haciendo que MiniZinc haga una búsqueda lexicográfica de tal forma que devuelva el resultado final en una sola iteración.

El siguiente modelo incluye estas dos posibles mejoras:

```
include "global_cardinality.mzn";

int:Q;
array [1..Q] of int: f;
int:R;
array [1..R] of int: c;
int:level;

int:people = sum(i in 1..Q)(f[i]);
int:maxf = max([f[i] | i in 1..Q]);
int:maxc = max([c[j] | j in 1..R]);
int:a_bound = min([maxf,maxc]);

array [1..level] of var 0..people:anom;

array [1..Q*R] of var 0..a_bound:aPlain; % a now must be a plain vector

% C1 initial domain
constraint forall([aPlain[i]>=0| i in 1..Q*R]);

% C2 every person is appointed
constraint forall([ sum([aPlain[(i-1)*R+j] | j in 1..R])==f[i] | i in 1..Q]);

% C3 take into account the maximum capacity of each resource
constraint forall([ sum([aPlain[(i-1)*R+j] | i in 1..Q])<=c[j] | j in 1..R]);

% C4
constraint global_cardinality(aPlain,[i | i in 1..level],anom);

solve :: int_search(aPlain, input_order, indomain_min, complete) satisfy;
```

De todas formas, hemos visto que estas posibles mejoras no redundan en mejores resultados en los experimentos, por lo que hemos empleado la primera versión para obtener los resultados experimentales del siguiente capítulo.

5 | Experimentos

5.1 Mejora de anonimato

El objetivo de este capítulo es comparar el nivel de anonimato obtenido con la asignación de citas aleatorias y la generada mediante nuestro programa MiniZinc. Para establecer esta comparación se hablará de la distancia entre dos vectores de anonimato, definida como la diferencia de las posiciones que ocupan en el orden lexicográfico contando desde el mayor valor.

Por ejemplo, dada una población de 4 personas, se pueden generar hasta 5 posibles vectores de anonimato:

1. (4, 0, 0, 0)
2. (2, 1, 0, 0)
3. (1, 0, 1, 0)
4. (0, 2, 0, 0)
5. (0, 0, 0, 1)

donde el menor anonimato es (4, 0, 0, 0) que indica que las 4 personas tienen cuasi-identificadores distintos, mientras que el (0, 0, 0, 1) es el mejor, al implicar que todos los usuarios tienen el mismo cuasi-identificador (un 1 en la posición 4, significa que hay un solo cuasi-identificador, que se repite 4 veces). La distancia, por ejemplo entre (0, 2, 0, 0) y (2, 1, 0, 0) es $d((0, 2, 0, 0), (2, 1, 0, 0)) = 4-2 = 2$.

Se destaca que este concepto de distancia no verifica la propiedad simétrica, y no es, por tanto, una distancia en el sentido matemático sino una distancia dirigida. Se cumple que dados dos vectores de anonimato, v1 <= v2 si y solo si $d(v1,v2) >= 0$.

Para la mejora de anonimato se parte de 3 valores:

- *vBase*: vector de anonimato que se tiene antes de incluir las citas, es decir teniendo en cuenta solo los cuantificadores iniciales:
- *vRandom*: vector de anonimato que se tiene tras incluir las citas generadas aleatoriamente, incorporando las columnas que señalan la cita.
- *vPR*: vector de anonimato obtenido con programación con restricciones

Es fácil comprobar que el anonimato es monótono decreciente en cuanto al número de cuasi-identificadores: si a una población dada le incorporamos nuevos cuasi-identificadores tendremos un vector de anonimato igual o peor. Tenemos por tanto que *vRandom*>=*vPR*>=*vBase* (recordemos que mayor vector, menor anonimato).

Para estudiar la mejora vamos a definir la cantidad (*vRandom*-*vPR*) / (*vBase*-1)

Esta es una cantidad entre 1 y -1 que nos indicará el grado de mejora logrado al comparar *vRandom* y *vPR*.

5.2 Descripción de los experimentos

Se han realizado pruebas para los siguientes distintos tamaños de poblaciones y para cada población se han considerado dos casos extremos:

1. Población de tamaño 10
    - Primer caso:
      Una población de mujeres entre 55 y 60 años con 2 códigos postales y sin profesión.
      En cuanto a los recursos, se dispone de 5 recursos con capacidades que varían entre 1 y 3. También se han especificado como atributos que representan los cuasi-identificadores edad y CP.
    - Segundo caso:
      Una población de mujeres entre 55 y 60 años con 3 códigos postales y sin profesión.
      En cuanto a los recursos, se dispone de 3 recursos con capacidades en 3 y 4. También se ha especificado como atributo que representa el cuasi-identificador, edad.

2. N = 20
    - Primer caso:
      Una población de mujeres entre 55 y 60 años con 2 códigos postales y sin profesión.
      En cuanto a los recursos, se dispone de 7 recursos con capacidades que varían entre 2 y 3. También se han especificado como atributos que representan los cuasi-identificadores edad y CP.
    - Segundo caso:
      Una población de mujeres entre 55 y 60 años con 3 códigos postales y sin profesión.
      En cuanto a los recursos, se dispone de 4 recursos con capacidades en 4 y 6. También se ha especificado como atributo que representa el cuasi-identificador edad.

3. N = 30
    - Primer caso:
      Una población de mujeres entre 55 y 60 años con 2 códigos postales y sin profesión.
      En cuanto a los recursos, se dispone de 10 recursos con capacidades que varían entre 3 y 4. También se han especificado como atributos que representan los cuasi-identificadores edad y CP.
    - Segundo caso:
      Una población de mujeres entre 55 y 60 años con 3 códigos postales y sin profesión.
      En cuanto a los recursos, se dispone de 5 recursos con capacidades en 5 y 7. También se ha especificado como atributo que representa el cuasi-identificador edad.

A continuación se mostrarán los detalles de 5 pruebas de las 20 generadas para cada uno de los casos anteriores, habiendo sido elegidas por su disparidad entre sus datos.

Población N = 10	Caso	Vector aleatorio	Vector anonimato	Mejora de anonimato	Tiempo de ejecución
Prueba 1	Primer caso	(8, 1)	(5,1,1)	0,7999	2´35 segundos
	Segundo caso	(6,2)	(2,2,0,1)	0,8125	2´35 segundos
Prueba 2	Primer caso	(8,1)	(2,1,2)	0,9411	2´35 segundos
	Segundo caso	(8,1)	(1,3,1)	0,9545	2´35 segundos
Prueba 3	Primer caso	(10)	(3,2,1)	0,9999	2´35 segundos
	Segundo caso	(4,3)	(2,2,0,1)	0,5625	2´35 segundos
Prueba 4	Primer caso	(6,2)	(3,2,1)	0,7272	2´35 segundos
	Segundo caso	(5,1,1)	(3,2,1)	0,5	2´35 segundos
Prueba 5	Primer caso	(10)	(4,3)	1	2´35 segundos
	Segundo caso	(2,4)	(0,0,2,1)	0,5946	2´35 segundos

Tabla 11

Al ser una población tan pequeña no hay diferencia en el tiempo de ejecución en ambos casos, en cambio sí que se puede apreciar la mejora de anonimato al realizar la comparación entre los vectores aleatorio y anonimato.

Como se ha explicado en el apartado 3.2 y tomando en cuenta los datos de las columnas vector aleatorio y vector anonimato, se concluye que el vector anonimato generado gracias a MiniZinc es mejor que el vector aleatorio debido a que, cogiendo por ejemplo el segundo caso de la primera prueba, se identifican a 4 personas menos que si se tomara en cuenta el vector aleatorio.

Así mismo, utilizando otra vez el ejemplo anterior, se observa una mejora del anonimato de 0,8125 (81’25%) con respecto al vector aleatorio.

Nótese que en el primer caso de la quinta prueba, comparando el vector aleatorio con el vector anonimato se ha conseguido una mejora del 100% (1), es decir, se pasa de desvelar la identidad

de toda la población, a desvelar solo al 40% de la población, teniendo en cuenta que el vector aleatorio es el peor vector posible que se puede conseguir.

Población N = 20	Caso	Vector aleatorio	Vector de anonimato	Mejora de anonimato	Tiempo de ejecución
Prueba 1	Primer caso	(14,3)	(3,4,3)	0,7331	80 segundos
	Segundo caso	(14,3)	(0,2,1,2,1)	0,9868	3 segundos
Prueba 2	Primer caso	(12,4)	(4,5,2)	0,5208	62 segundos
	Segundo caso	(5,3,3)	(0,3,2,2)	0,7051	2’85 segundos
Prueba 3	Primer caso	(16,2)	(4,5,2)	0,9210	224 segundos
	Segundo caso	(4,8)	(0,3,3,0,1)	0,6505	2’85 segundos
Prueba 4	Primer caso	(20)	(4,5,2)	0,9999	300 segundos
	Segundo caso	(4,8)	(1,2,2,1,1)	0,5465	3 segundos
Prueba 5	Primer caso	(15,1,1)	(5,6,1)	0,9489	247 segundos
	Segundo caso	(13,2,1)	(1,0,2,2,1)	0,9759	2’85 segundos

Tabla 12

Al aumentar la población, se incrementa el tiempo de ejecución, y así mismo se observa una gran diferencia entre los tiempos del primer caso y del segundo caso, mostrados en la *Tabla 12*, debido a, no sólo que el primer caso tiene otro atributo más como representante del cuasi-identificador, sino también se disponen de más recursos, pero con menos capacidades que en el segundo caso.

Todo esto conlleva a que las poblaciones de los primeros casos de las pruebas estén más dispersadas con respecto a las poblaciones de los segundos casos en cuanto a la asignación de citas.

Tomando como ejemplo la prueba 4, debido a la gran diferencia entre los datos de sus casos, se observa una diferencia de mejora respecto a los vectores de aleatorio y anonimato; en el primer caso del 0,9999 (99,99%) y en el segundo caso del 0,5465 (54,65%) además de un tiempo de ejecución muy dispar, que pasa de los 5 minutos que tardó el primer caso, a los escasos 3 segundos que tardó el segundo.

Población N = 30	Caso	Vector aleatorio	Vector de anonimato	Mejora de anonimato	Tiempo de ejecución
Prueba 1	Primer caso	(28,1)	(3,1,3,4)	0,9996	360 segundos
	Segundo caso	(10,7,2)	(1,0,1,1,3,0,1)	0,8714	3,95 segundos
Prueba 2	Primer caso	(26,2)	(0,7,4,1)	0,9862	120 segundos
	Segundo caso	(10,10)	(0,0,2,2,2,1)	0,8992	3,95 segundos
Prueba 3	Primer caso	(24,3)	(3,5,3,2)	0,9963	148 segundos
	Segundo caso	(12,9)	(0,0,4,2,2)	0,9383	4 segundos
Prueba 4	Primer caso	(20,5)	(0,5,4,2)	0,9935	324 segundos
	Segundo caso	(18,3,2)	(0,1,1,0,1,1,2)	0,9887	3,86 segundos
Prueba 5	Primer caso	(22,4)	(2,4,4,2)	0,9784	227 segundos
	Segundo caso	(14,8)	(0,1,0,2,0,1,2)	0,9661	4 segundos

Tabla 13

En el caso de la población de tamaño 30 se consigue una gran mejora de anonimato, puesto que en casi todas las pruebas se ha mejorado tanto el k-anonimato como el vector de anonimato, pero el tiempo de ejecución se ha incrementado de una manera drástica con respecto a las otras poblaciones.

A partir de los datos del primer caso de la prueba 1, se puede concluir que, a pesar de que el vector anonimato sigue teniendo un 1-anonimato, igual al k-anonimato del vector aleatorio, la mejora es del 0,9996 (99,96%) debido a la gran diferencia entre ambos vectores de anonimato, pasando de desvelar la identidad de hasta 28 personas, a identificar solo 3.

En el primer caso de la prueba 4, no solo se mejora el vector de anonimato un 0,9935 (99,35%) sino que también se mejora el k-anonimato, pasando de 1-anonimato a 3-anonimato.

6 | Implementación

Para almacenar la información generada sobre la población, se hace uso de una base de datos relacional PostgreSQL [8].

PostgreSQL es gratuito y libre que permite desarrollar bases de datos relacionales robustas y eficientes, además de que ofrece una gran cantidad de opciones avanzadas. De hecho, es considerado el motor de base de datos más avanzado en la actualidad, además de aportar mucha flexibilidad a diferentes proyectos. Por ejemplo, permite definir funciones personalizadas por medio de varios lenguajes, en este caso utilizando PL/Java.

Otra ventaja de PostgreSQL es que está disponible para muchas plataformas y ofrece el código fuente desde el sitio oficial como Mac OS X. Windows y Ubuntu.

Así mismo, PostgreSQL ofrece la herramienta oficial pgAdmin para administrar sus bases de datos, dando la opción también de administrar las bases de datos mediante línea de comandos.

Para el desarrollo software de este proyecto se ha empleado Java como lenguaje de programación principal utilizando el entorno de programación Eclipse.

Se ha optado por Eclipse por ser uno de los principales entornos de programación que se ha utilizado durante la carrera, debido a ser código abierto y multiplataforma además de su interfaz intuitiva y sencilla.

Así mismo, se ha empleado también MiniZinc, explicado en el tema 4, y se ha hecho uso de Bash Shell para unificar todos los programas que forman este proyecto y calcular cuánto tiempo tarda en ejecutarse el proyecto completo.

La implementación de este proyecto se dividido en distintos proyectos de Java, cada uno con una o varias funciones asignadas.

Para la correcta ejecución del primer programa, se debe asignar un valor a todos los campos requeridos. Sin embargo, si no se han rellenado correctamente los campos, se interrumpirá el programa mostrando un mensaje de error informando sobre la causa del problema.

Para la conexión con la base de datos PostgreSQL del usuario mediante el conector JDBC, los primeros campos que se solicitan son la url de la base de datos, el usuario y la contraseña de la misma.

Para la creación de los datos ficticios, se debe especificar primero el nombre de la tabla en la que se guardará la información, el número de personas que se quiere estudiar, así como el rango de edades, es decir, edad mínima y máxima de los pacientes que se van a examinar; también se incluye la cantidad de códigos postales para determinar la zona en la que vive la población el género de la población, representados por: 0 es mujer, 1 es hombre y 2 se refiere a ambos; y el número de profesiones, que representa la actividad que desempeña cada individuo. Además, el usuario debe indicar la cantidad de recurso, que representa la hora, día y hospital de la cita s de los que dispone y especificar el rango de capacidad de cada uno, para que se asigne aleatoriamente una capacidad para cada recurso, estando dentro de los límites del rango introducido.

Así mismo el programa automáticamente generará otra tabla en la que se reflejan el número de recursos y la capacidad de cada uno, a partir del nombre de la tabla indicado.

Si la capacidad máxima total de todos los recursos es menor que el número de pacientes introducidos, se ajustarán las capacidades de los recursos necesarios para cubrir a todos los pacientes hasta el máximo indicado.

Cabe destacar, que, si el nombre de las tablas a crear ya existía con anterioridad, se eliminarán de la base de datos para evitar la sobre escritura.

En el segundo programa se realiza un nuevo acceso a la base de datos del usuario comprobando que las tablas ya se han creado y que contienen información. El objetivo de este proyecto es realizar una asignación aleatoria de los recursos teniendo en cuenta sus capacidades, actualizando el campo recurso aleatorio de la tabla de la población.

Durante el flujo de trabajo de este proyecto se ha querido saber el valor de k-anonimato de la población creada, que dependerá de los atributos indicados por el usuario que representan los cuasi-identificadores. Para ello se ha llevado a cabo la creación del tercer programa que muestra por pantalla dicho nivel de anonimato.

En el cuarto programa, se procede a crear el fichero con extensión dzn para la posterior ejecución del código MiniZinc.

Para ello se reutilizará los atributos asignados por el usuario en el programa tres. Además, se reconectará a la base de datos de PostgreSQL para realizar los cálculos necesarios para escribir los siguientes datos en el fichero:

- Q: número de cuasi-identificadores diferentes.
- f: vector que representa la frecuencia de cada cuasi-identificador, es decir, el número de veces que se repite cada cuasi-identificador.
- R: su valor representa el número de recursos de los que se dispone.
- c: vector de capacidades de cada recurso.

A continuación, se procede a la ejecución del quinto programa que incluye las siguientes funciones:

- Acceder a MiniZinc mediante una llamada a la línea de comandos (cmd), utilizando el fichero con extensión mzn, explicado en el tema 4 y el fichero .dzn generado anteriormente. Para generar una nueva asignación de citas.

  Un ejemplo de fichero .dzn se puede observar en la imagen 1 en el apartado 4.2 *El Modelado.*

El programa accederá a MiniZinc hasta que se hayan asignado citas a toda la población. Esto se consigue mediante la declaración de las siguientes variables:

- p = $\sum_{i=1}^{Q}$ f[i] Es la población total a examinar.
- l = 1 El siguiente nivel del vector de anonimato a resolver.
- v = [] Vector de anonimato, inicialmente con 0 componentes.

- Leer los resultados generados por MiniZinc y escribirlos en el fichero dzn de tal manera que se pueden ver los distintos vectores de anonimato generados.

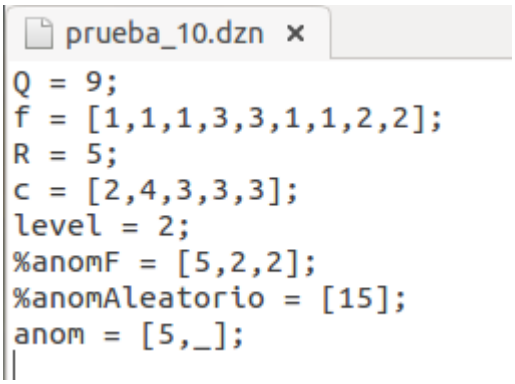

Imagen 2

La asignación de citas se da por finalizada una vez que la suma de los valores del vector v multiplicados por su índice es igual a la población total ($\sum_{n=1}^{l-1}$ v[n] $*$ $n < P$), mientras que no se cumpla esta condición, el programa aumentará el vector *v* además de insertar en la última posición de *v* (*v[l]*), el anonimato generado por appointment.mzn. Así mismo, se incrementará el siguiente valor de *l* a resolver.

- Utilizando los datos generados por el programa MiniZinc, se calcula la distancia entre los vectores para demostrar si se ha procedido a mejorar o empeorar el nivel de anonimato.

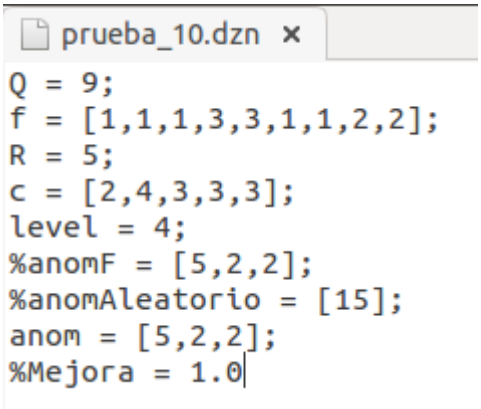

Imagen 6

Como se ha mencionado al principio de este tema, se ha implementado un pequeño programa en Shell para ejecutar los programas anteriores en serie. De esta manera, se pueden crear varios modelos de población con los mismos datos introducidos por el usuario final, obteniendo diferentes resultados.

Para la creación de los modelos de población se pide al usuario introducir el número de pruebas que desea realizar además de todos los parámetros requeridos para la correcta ejecución de los programas explicados anteriormente.

Si se quiere calcular el tiempo que se tarda en generar cada experimento, se puede utilizar a la vez el comando *time* [10], que mostrará al finalizar la ejecución los siguientes tiempos:

- el tiempo real transcurrido entre la llamada y la finalización de orden.
- el tiempo de usuario del procesador (tms_utime + tms_cutime).
- el tiempo de sistema del procesador (tms_stime + tms_cstime.)

7 | Conclusiones y trabajo futuro

Hemos realizado un sistema de citas que garantiza el mejor anonimato posible utilizando el concepto de k-anonimato.

Durante el proyecto hemos comprobado que el concepto de k-anonimato no es suficiente y por eso utilizamos un concepto ampliado, el vector de anonimato.

Como se ha mencionado anteriormente, para conseguir el k-anonimato utilizamos en un primer momento el resolutor base minizinc, pero como el tiempo de ejecución era demasiado alto, optamos por el resolutor mzn-gecode, que disminuyó notablemente dicho tiempo. Aun así, al aumentar el tamaño de la población, el tiempo de ejecución también se incrementa de forma significativa.

Por el motivo anterior, podemos concluir que el cuello de botella de nuestro proyecto es MiniZinc.

Dicha herramienta es el cuello de botella de nuestro proyecto, es decir, cada vez que la población es mayor, el resolutor incrementará el tiempo de ejecución, por lo que, como se observa en el apartado 5.2, el tiempo aumenta de forma significativa.

La programación con restricciones solo es válida para poblaciones pequeñas, sin embargo, da una medida del menor vector de anonimato que se puede emplear para comprobar la bondad de otros métodos de asignación aplicables a poblaciones de tamaño real.

Durante los experimentos realizados, hemos llegado a las siguientes conclusiones:

- Verificamos que al comparar el vector de anonimato aleatorio y el generado por Programación con Restricciones, no siempre se obtiene una mejora desde el punto de vista del k-anonimato, ya que esto depende no solo del número de valores de cuasi-identificadores, sino también del número de recursos de los que se dispone.
- Aunque el k-anonimato no varíe, eso no significa que el vector aleatorio y el vector anonimato sean iguales, es decir, se puedan identificar menos personas en el vector anonimato frente al vector aleatorio.
- Se puede conseguir casi el 100% de mejora comparando los vectores aleatorio y anonimato sin que varíe el k-anonimato, en el caso de que ambos vectores correspondan con el mismo k.
- Una manera de aumentar la distancia entre el vector aleatorio y el vector anonimato, es aumentando la longitud del vector de anonimato, es decir, dispersar más la población a la hora de asignar las citas.

Hay que notar que otros conceptos de anonimato como *l*-Diversity [5] y *t*-Closeness [6] no son aplicables en el momento de la asignación de citas porque dependen del valor final de los resultados del test de screening, mientras que nosotros estamos en el contexto de las citas, cuando aún no es visible el resultado final. Sin embargo, estos conceptos sí serán aplicables una vez acabado el test, y se beneficiarán de la asignación de citas inicial.

Como trabajo futuro estaría desarrollar una nueva técnica de asignación de citas que soporte poblaciones más grandes, y que, sin alcanzar el mejor anonimato sí que mejore la asignación aleatoria realizada en la actualidad.

8 | Conclusions and future work

We have performed a system of appointments, which guaranteed the best possible anonymity using the concept of k-anonymity.

During the project we verified that the concept of k-anonymity is not sufficient, therefore, we have also used a broadened concept, anonymity vector (vector de anonimato).

As it is mentioned before, at the beginning we used the base resolver of MiniZinc, but the runtime was too high to obtain the k-anonymity. For this reason, we chose using mzn-gecode solver which decreased notably the runtime. Despite that, the increasing of population is directly proportional with the runtime.

Because of the previous reasons, we conclude that the bottleneck of our project is MiniZinc.

This tool is the bottleneck of our project and that means, when the population increases, the runtime of the resolver will be higher, therefore, the time increases significantly as it is indicated in the section 5.2.

The constraint programming is only valid for a little population, in spite of that it gives the measurement of the smallest anonymity vector, which could be used to check the good quality of other assignment methods which are applicable to a real population.

While we were doing the experiments, we got the following conclusions:

- We verified that the k-anonymity does not always improve because it does not only depend on the numbers of the values of the quasi-identifiers, but it also depends on the number of available resources.
- When the k-anonymity is invariable, it does not mean that the random vector and the anonymity vector are the same, that is to say, we can disclose less identities of people from the anonymity vector than from the random vector.
- We could obtain about 100% of the improvement if we compare the random and anonymity vectors without changing k-anonymity, when both of them have the same k.
- One of the options to increase the distance between the random vector and the anonymity vector is by increasing the length of the anonymity vector, that is to say an appointment scheduling with more population dispersion.

It has to be noted that there are other anonymity concepts such as l-Diversity [5] y t-Closeness [6] but they are not applicable because they depend on the final value of the screening, whereas we want to ensure certain anonymity by scheduling the appointments, when the tests results have not been revealed yet.

For future work, a new technique of the appointment assigning could be developed, to cover a considerable population without approaching the best anonymity improvement, but it has to improve the random appointment assigning, which we have done.

9 | Bibliografía

[1] P. Samarati, Protecting respondents identities in microdata release, Knowledge and Data Engineering, IEEE Transactions on (Volume:13 , Issue: 6), 2000

[2] L. Sweeney. *k*-anonymity: a model for protecting privacy. International Journal on Uncertainty, Fuzziness and Knowledge-based Systems, 10 (5), 2002; 557-570.

[3] Kimbal Marriott and Peter Stuckey, Programming with Constraints: An Introduction. ISBN: 9780262133418, February 1998.

[4] Nethercote, Nicholas and Stuckey, Peter J. and Becket, Ralph and Brand, Sebastian and Duck, Gregory J. and Tack, Guido, MiniZinc: Towards a Standard CP Modelling Language. ISBN: 978-3-540-74969-1, pages 529-543, 2007.

[5] Ashwin Machanavajjhala, Johannes Gehrke, Daniel Kifer and Muthuramakrishnan Venkitasubramaniam. *l*-Diveristy: Privacy Beyond *k*-Anonimity. Journal ACM Transactions on Knowledge Discovery from Data (TKDD) on (Volume: 1, Issue: 1, Article No. 3), March 2007.

[6] Ninghui Li, Tiancheng Li and Suresh Venkatasubramanian. *t*-Closeness: Privacy Beyond *k*-Anonymity and *l*-Diversity. In Proc. of IEEE 23rd Int'l Conf. on Data Engineering (ICDE'07), 2007.

[7] http://lbe.uab.es/vm/sp/old/docs/prevencion/cribados.pdf

[8] https://platzi.com/blog/que-es-postgresql/

[9] http://www.MiniZinc.org/downloads/doc-latest/MiniZinc-tute.pdf

[10] http://francisconi.org/linux/comandos/time

[11] http://medical-dictionary.thefreedictionary.com/screening